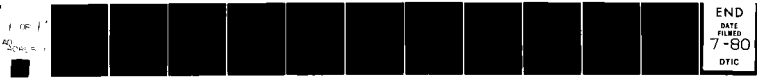
AD-A085 517 VIRGINIA POLYTECHNIC INST AND STATE UNIV WASHINGTON --ETC F/G 9/2
CONVERTING DEC-10 SIMULA PROGRAMS TO CMS SIMULA.(U)
JUL 79 R J ORGASS AFOSR-79-0021
UNCLASSIFIED VPI/SU-TM-79-7 AFOSR-TR-80-0447 NL



EXTENSION DIVISION

VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE
GRADUATE PROGRAM IN NORTHERN VIRGINIA

P. O. Box 17186
Washington, D. C. 20041
(703) 471-4600

CONVERTING DEC-10 SIMULA PROGRAMS TO CMS SIMULA*†

Richard J. Orgass

Technical Memorandum No. 79-7

July 1979

ABSTRACT

A collection of advice and directions for moving SIMULA programs is given.

---

* Work supported in part by the Air Force Office of Scientific Research, Air Force Systems Command, under Grant No. AFOSR-79-0021.

† The information in this document is subject to change without notice. The editor, the authors, Virginia Polytechnic Institute and State University, the Commonwealth of Virginia and the United States Government assume no responsibility for errors that may be present in this document or in the programs described here.

This note reports on the writer's experience moving DEC-10 SIMULA programs to CMS and executing the programs with CMS SIMULA. With minor exceptions described below, the languages implemented by the two compilers is the same and this makes the transfer of SIMULA programs very easy. In contrast, it is very difficult to transfer Fortran programs from a DEC-10 to CMS.

The two main sources of difficulty in transferring programs are the differences in the character set and file system of the two systems and the fact that CMS lacks many of the capabilities that TOPS-10 users take for granted. The directions for moving DEC-10 SIMULA programs to IBM SIMULA that are provided with release 3 refer to older versions of IBM SIMULA and it is now possible to preserve most of the separately compiled classes and procedures while moving programs to CMS SIMULA.

The discussion that follows assumes that programs running under release 3 of DEC-10 SIMULA are to be moved to CMS SIMULA, Version 7.00, from Imperial College of Science and Technology. Earlier versions require some additional work as described below.

1. Changes to the Program Text

This section contains information that is useful when preparing DEC-10 SIMULA for transfer to CMS SIMULA. Some of this information was obtained from Part III of the DEC-10 SIMULA Handbook and other items were learned by trial and error. The procedure suggested might best be characterized by the statement "I wish I knew enough to do it this way."

There are a few differences in the syntax of SIMULA in the two implementations. Since the particular characters also cause problems when transferring files, it is best to make changes on the DEC-10. The following changes to the source text should be made:

(1) Replace all occurrences of square brackets, [and], with parentheses, (and). [The square brackets in text and character constants might just as well be changed to parentheses becauses they will be transformed into other characters in the ASCII to EBCDIC translation. Editing will be needed in any case.]

(2) Replace all occurrences of \= with NE. IBM SIMULA uses ~= or NE and there are difficulties with ~ when moving text files.

(3) Replace all occurrences of \ with NOT.

(4) CMS SIMULA does not accept ! as an abbreviation for COMMENT. Replace all such occurrences with COMMENT.

(5) While the source files are in transit they will not have a name associated with them so it is a good idea to add a comment with the file name at the beginning of each file.

(6) CMS SIMULA does not concatenate text constants when they follow each other. If the program contains

```
"abc" "def"
```

replace this text with

```
"abcdef"
```

or

```
conc2("abc","def")
```

(7) The DEC-10 library procedure conc is not available in CMS. Replace calls to conc with nested calls to conc2. (It will be necessary to compile the SIMULA source for conc2 as given in Part III of the DEC-10 SIMULA Handbook.)

(8) Remove all line numbers from files and expand tabs. Here is an easy way to do this:

```
.r pip
*all:*.sim/w/x/n=all:*.sim
*^C
```

The files are still not ready for transfer. CMS SIMULA expects its input as 80 column records and reads only columns 1-72. (The whole CMS environment is directed to card images and the SIMULA compiler follows the conventions.) The easiest way to change the record length of your source files is to use SIMED. Version 7.00 of CMS SIMULA supports upper and lower case program text in the same way as the DEC-10 implementation so it is not necessary to modify the program text. If you are going to use an earlier version, all program text except comments and character and text constants must be in upper case; SIMED can perform this translation.

Although option statements are not used by the CMS compiler, it is best to leave them in the source text for modification on CMS. Some of the option switches are replaced by command lines in the source text and others are replaced by options in the command that invokes the compiler. It is quite straightforward to replace these option statements using CMS when all the documentation is immediately at hand.

The external compilation facility of CMS SIMULA is in some ways more general than the DEC-10 version and in other ways more

difficult to use because of the CMS file system. The DEC-10 statement

```
EXTERNAL CLASS mumble=FOO.SIM;
```

should be replaced by the statement

```
EXTERNAL CLASS mumble=FOO;
```

It is not necessary to have external statements in all parts of a program that implicitly reference other separately compiled parts of a program. This means that many of the external statements in DEC-10 SIMULA programs can be removed. Retain the innermost external statement (that is, in the earliest program part to be compiled) and delete all of the copies in containing (or later compiled) parts of the program. External declarations obey the usual scope rules in CMS SIMULA. The order of compilation rules of the DEC-10 implementation apply to the CMS implementation but the compiler does not do all the checking that is done by the DEC-10 compiler.

Some aspects of separate compilation of classes are more difficult in CMS. If your program text (in file foo) is of the form,

```
EXTERNAL CLASS mumble;
mumble CLASS foo;
BEGIN
    ...
END of foo;
```

the class mumble must be compiled using the CMS command

```
simula mumble (extern c
```

and then the class foo is compiled using the CMS commands:

```
filedef mumble disk mumble simclass
simula foo
```

On the other hand, if your program text (in file foo) is:

```
BEGIN
    EXTERNAL CLASS mumble;
    mumble CLASS foo;
    BEGIN
        ...
    END of foo;
    ...
END of program.
```

or

```
BEGIN
    EXTERNAL CLASS mumble;
    mumble BEGIN
        ...
    END of mumble block;
    ...
End of program.
```

then mumble is compiled as above but foo is compiled with the CMS command:

```
simula foo (class mumble
```

(The filedef command is not needed.)

External procedures are compiled with the extern p option and thereafter can be used as needed with external declarations. There is no level number associated with a separately compiled procedure or class.

In one respect, the separate compilation facility of CMS SIMULA is more restrictive. Using the DEC-10 compiler, it is possible to compile the following class

```
EXTERNAL CLASS video_file=VIDFIL.SIM;
CLASS slr_parser(f); REF(video_file)f;
BEGIN
    ...
END of slr_parser;
```

This class cannot be separately compiled in CMS SIMULA because the parameters of a class must be either a basic type of SIMULA (e.g., integer, character, text, etc.) or be declared in the same source file.

2. Transferring Files

The program CHANGE supplied by many TOPS-10 installation is a reasonably reliable way of writing files on tape for transfer from a DEC-10 to an IBM machine. A number of different options were tried and the procedure described below appears to be the least painful.

TOPS-10 users are accustomed to thinking of text files as a sequence of characters divided into records by <cr><lf>; this view is quite useless in CMS. The CMS file system views text files as a sequence of records, each of a fixed length called LRECL. This writer wasted a great deal of time trying to avoid this conclusion and strongly recommends adopting the CMS view of a file in spite of an enormous number of unnecessary blanks.

CHANGE should be used to write SIMULA source files (reformatted as described above) on an unlabeled ASCII tape. The files should be written as fixed length records with 80 column records in 8-bit ASCII with high order bit 0. The ASCII to EBCDIC translation performed by CHANGE has many strange properties which include mapping lower case letters into upper case and confusing special characters. It's a good idea to write several repetitions of the file sequence on the tape in case of problems with the tape -- different brands of drives will write and read the tape and this aggravates the usual problems.

Tapes written in this way can be read using IBEDIT. A CMS EXEC for reading the tapes can be obtained from the author.

There are a few annoying properties of the ASCII to EBCDIC translation performed by IBEDIT: (1) There are two EBCDIC codes for each curly bracket ({ and }). IBEDIT selects the code which is not printed by the TN print train and which is not the code selected by the translate table used to support ASCII terminals. (2) There are three EBCDIC characters that correspond to vertical line (ASCII 124) and IBEDIT choses one that is not printed by the TN train or on an ASCII terminal. (3) A similar statement applies to ~ (ASCII 126). (4) The backslash (\, ASCII 92) is translated into a completely unprintable character.

A CMS SIMULA program to correct this translation may be obtained from the author.

3. Programming Problems

CMS SIMULA imposes the Common Base restriction on text parameters passed by name while the DEC-10 implementation is more liberal. The following sketch of a procedure will execute correctly in DEC-10 SIMULA when called with a text constant as actual parameter:

```
PROCEDURE mumble(t); NAME t; TEXT t;
BEGIN
    ...
    x :- Copy(t);
    i := t.length
    y := t.sub(1,4)
    ...
END of mumble;
```

In CMS SIMULA, a run time error will occur. An ugly but easy

patch will avoid this problem:

```
PROCEDURE mumble(a); NAME a; TEXT a;
BEGIN
    TEXT t;
    <declarations>
    t :- Blanks(200);
    t := a;
    t :- t.strip;
    <remainder of procedure body as above>
END of mumble;
```

The CMS environment imposes severe restrictions on interactive programs which must be dealt with in the program. Note that these are CMS restrictions which are enforced for all programs and SIMULA must comply.

In CMS, an empty line is an end-of-file. Therefore, there is no easy way to distinguish between an empty line of input and ^Z. This means that the code

```
Sysin.Inimage;
    IF Sysin.Image.Strip = NOTEXT
        THEN...
        ELSE...
```

will not work unless the user types one or more blanks followed by <cr>; it's very easy to forget the blank. If the blank is forgotten,

```
Sysin.Image.Sub(1,2) = "/*"
```

is true and the next read will cause an error termination.

The need for spaces when going through a long sequence of questions where the default answer is selected by an empty line is, at best, annoying and often leads to unexpected error terminations. This writer found it useful to treat end-of-file as an empty line. This requires reopening the terminal file:

```
i := Sysin.Image.Length;
Sysin.Close;
Sysin.Open(Blanks(i));
```

In TOPS-10, a file specification, e.g., STDN:SIMED.DAT[1000,237], completely describes a file in all places and at all times. One can read a file specification from the terminal and then open the file:

```
Outtext("Input File: ");
Breakoutimage;
Inimage;
input_file :- NEW Infile(Sysin.Image.Strip);
input_file.Open(Blanks(n));
```

If something goes wrong, SIMDDT will provide for recovery.

In CMS, the file specification, e.g., SIMED DATA A identifies a file only in the user's directory. This name cannot be used to directly read the file; a second name, called the DDNAME is needed! A partial solution to this problem is the following code:

```
EXTERNAL ASSEMBLY PROCEDURE cmscommand;
...
Outtext("Input File:");
Outimage; Outimage;
dd_name :- get_dd_name;
filedef :- conc2("FILEDEF ",dd_name);
filedef :- conc2(filedef, "DISK ");
filedef "- conc2(filedef, Sysin.Image.Strip);
cmscommand(filedef,return_code);
IF return_code = 0
    THEN BEGIN
             input_file:-NEW Infile(dd_name);
             input_file.Open(Blanks(n));
         END;
    ELSE < obtain correct file specification>
```

This code will work if the file exists but there will still be a run time error termination if the file does not exist. A second call on cmscommand using the STATE command of CMS can check on the existence of the file.

In the above discussion, get_dd_name is a text procedure that returns a unique valid DDNAME at each call.

The important thing to notice is that there is no error recovery via SIMDDT; the program must catch all errors. There are further details concerning LRECL and RECFM parameters for the FILEDEF command that are not discussed here. The CMS Command and Macro Reference Manual has a clear discussion.

TOPS-10 users will find this very crude and complicated. Please don't blame SIMULA -- this is CMS. The input/output procedures in CMS SIMULA are far better than those available in Fortran and PL/I. It is fairly straightforward to read files from SIMULA programs while it is impossible to read the same files from PL/I or Fortran programs!

Finally, a warning. In all IBM SIMULA versions before 7.00, the system procedure Letter treats only upper case letters as letters. In these earlier versions, the expression Letter('a') has the value FALSE.

4. Additional Help

The author has written a number of programs to facilitate the transfer of SIMULA programs from a DEC-10 to CMS. The properties of these programs are summarized below and more information may be obtained by writing to the author.

A technical memorandum, Transferring Files to an IBM Machine, gives directions for preparing files for transfer and for writing tapes. Source listings of APLSF and SAIL programs that are described in this memo are available. The programs are not available in machine readable form.

A CMS EXEC to read unlabeled ASCII tapes via an MVS batch job has been used extensively. It is a bit awkward to use but performs the job quite satisfactorily. A listing of the EXEC and a brief help file are available.

A CMS SIMULA program to correct the ASCII to EBCDIC translation performed by IBEDIT together with a help file is available. Paper copy or a CMS dump tape are available.

The DEC-10 SIMULA program SIMED has been modified for use with CMS SIMULA. Paper copy or a CMS dump tape are available.

A SIMULA class DIALOG that deals with the problems described above and also contains many of the procedures from the DEC-10 SIMULA library is available. The documentation includes a technical memorandum and a help file. The code is available on paper or a CMS dump tape.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1 REPORT NUMBER: AFOSR-TR- 80-0447	2 GOVT ACCESSION NO. (AD-A0[illegible]) 3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle): CONVERTING DEC-10 SIMULA PROGRAMS TO CMS SIMULA	5 TYPE OF REPORT & PERIOD COVERED: Interim
	6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s): Richard J. Orgass	8 CONTRACT OR GRANT NUMBER(s): AFOSR 79-0021
9 PERFORMING ORGANIZATION NAME AND ADDRESS: Virginia Polytechnic Inst. & State University, Department of Computer Science, Washington, DC 20041	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A2
11 CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research/NM, Bolling AFB, Washington, DC 20332	12 REPORT DATE: July 1979 13 NUMBER OF PAGES: 10
14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15 SECURITY CLASS (of this report): UNCLASSIFIED 15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

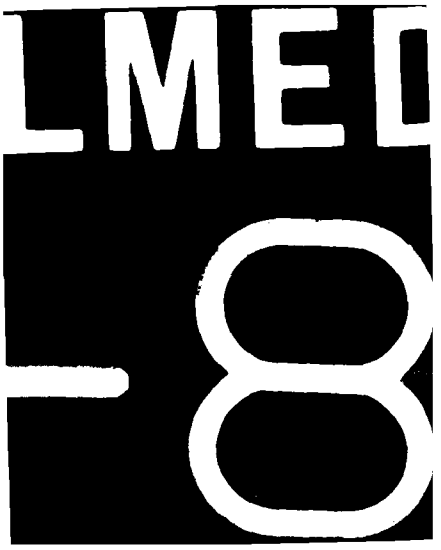
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A collection of advice and directions for moving SIMULA programs is given.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)


LMED
8